# Comunicacións industriais

## Práctica 03: Comunicación half-duplex entre un S7-200 e un PC

### Descrición da práctica:

Comunicación semiduplex (half-duplex) entre un PC no que programamos en CSharp unha aplicación para enviarlle un dato a un autómata S7-200 e recollelo de volta na mesma aplicación. Deste xeito traballamos con este tipo de comunicación propia dun protocolo RS-485. Dito protocolo é a base para entender outros de aplicación na industria como é o PROFIBUS.

### Coñecementos previos:

- Programación en CSharp
- Programación do modo Freeport, do manexo de interrupcións e dos punteiros en S7-200
- Coñecemento do funcionamento dos protocolos RS-232 e sobre todo RS-485

### Material necesario:

- PC con Microwin e Visual Studio 2017 instalado (CSharp)
- Autómata S7-200 con cable de programación PC/PPI

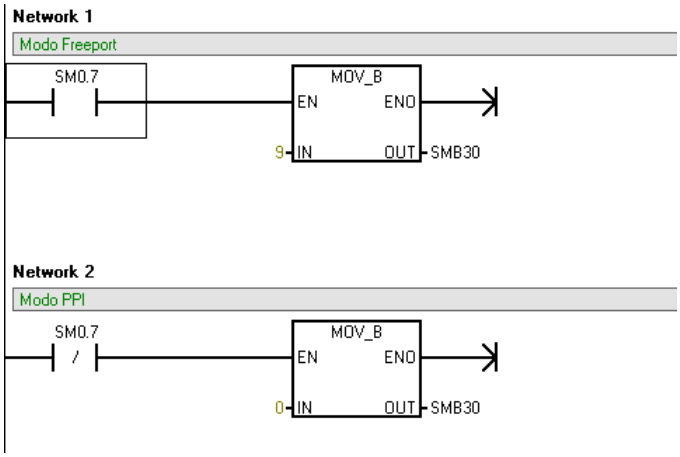
### Especificacións:

- Elaborarase un programa en Microwin no que habilitaremos o modo Freeport, as interrupcións para recoller caracteres polo porto serie e definiremos un punteiro para recoller máis de un byte cando entren polo porto do autómata.
- Programarase a interface gráfica en CSharp para xerar o dato a enviar ao PLC en formato palabra e para visualizar a resposta do autómata tamén en formato palabra.
- No programa de CSharp disporemos dos dous formularios que empregamos na práctica anterior, un principal no que se xera e se visualiza o dato na gráfica e un secundario para establecer os parámetros da comunicación.

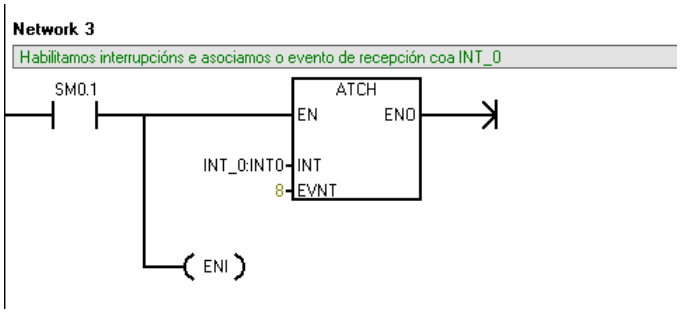
### Solución:

Empezamos polo programa do autómata.

Primeiro configuramos o modo Freeport

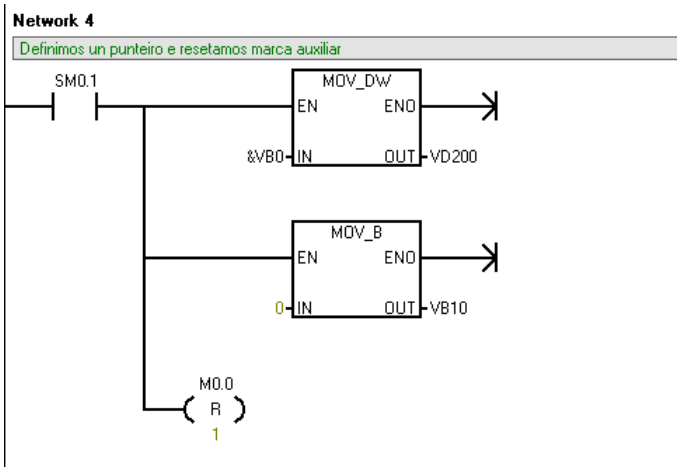



A continuación habilitamos as interrupcións no autómata e asociamos o evento de interrupción 8 (caracter recibido polo porto serie) coa subrutina de interrupción INT_0

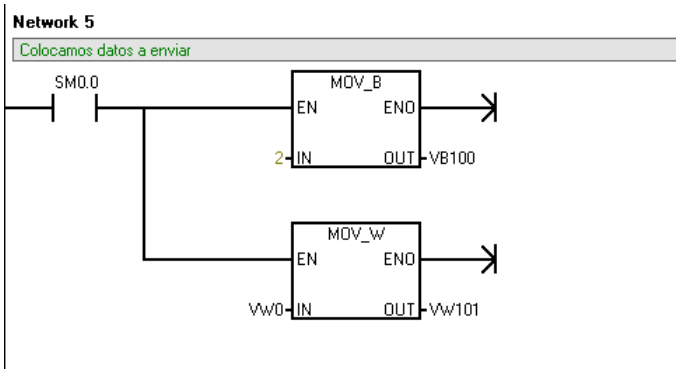



O concepto quizás máis complexo de entender nesta tarefa é o traballo con punteiros no autómata. Temos que definir unha posición de memoria (neste exemplo a VD200) que sirva para gardar outra posición de memoria (neste exemplo a VB0). O que dicimos tecnicamente que o punteiro (VD200) apunta á posición VB0.

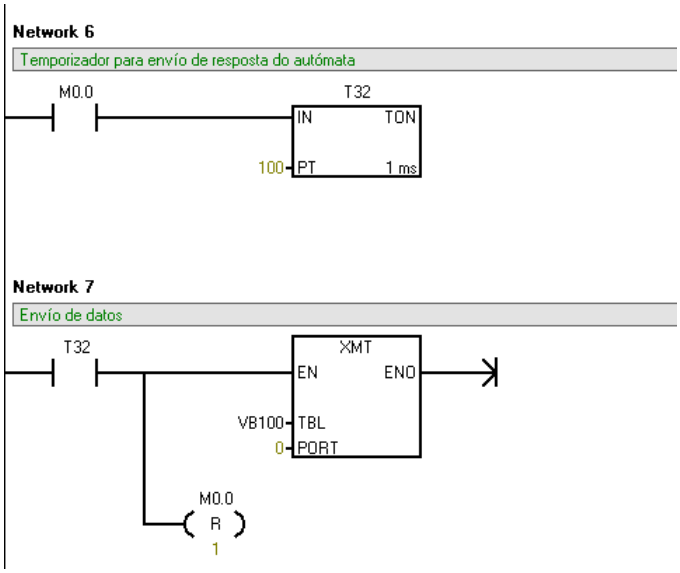
Despois faremos que o punteiro (VD200) vaia apuntando ás posicións sucesivas (VB1, VB2.....)




A continuación colocamos os datos que queremos retornarlle ao CSharp. Fixémonos que neste exemplo imos recoller a palabra procedente do PC na VW0 e retornámola desde a VW101.

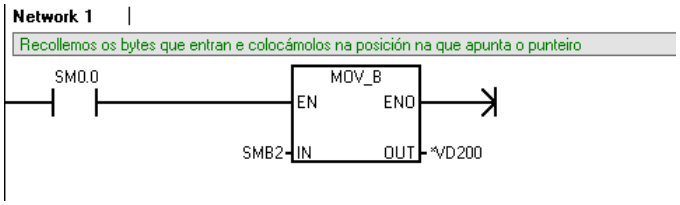



Por último, no programa principal devolvemos a palabra mencionada.

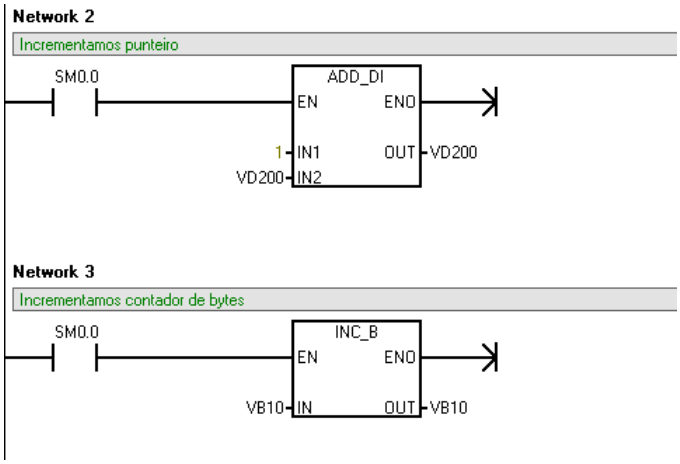



Na subrutina de interrupción INT_0 temos o seguinte:

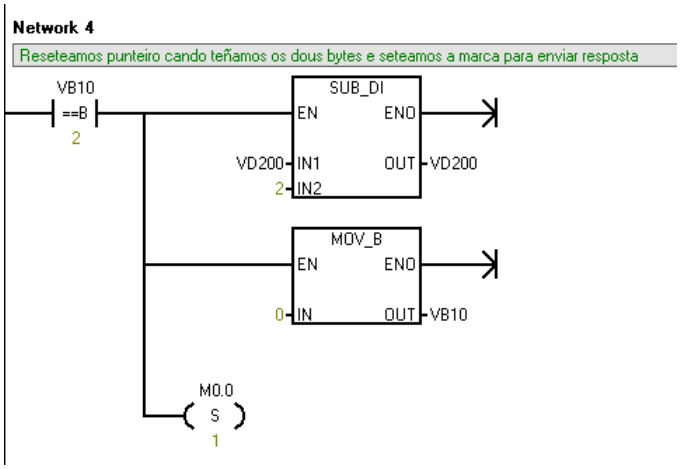
Primeiro recollemos os bytes que van entrando.




A continuación incrementamos o punteiro para que apunte á seguinte posición de memoria e incrementamos o contador de bytes para levar conta dos que entraron.

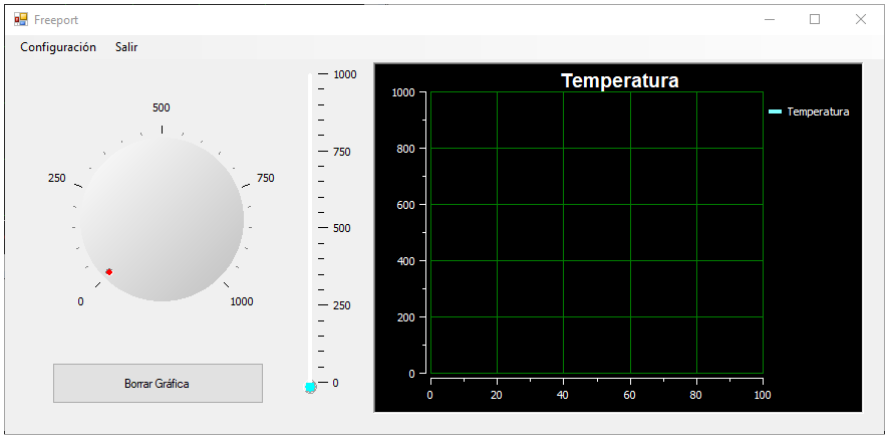



Por último, se entraron todos os bytes que se esperaban (neste exemplo dous), reseteamos o punteiro para que volva a apuntar á VB0, reseteamos o contador e activamos a marca auxiliar para esperar un tempo de 100ms (pode axustarse a un intervalo máis pequeno) antes de devolver a resposta ao PC.




O código do formulario de configuración da comunicación é o mesmo que o da práctica anterior, de feito pode aproveitarse o mesmo formulario e engadilo a este proxecto.

No programa principal teremos unha interface similar á seguinte.




O código sería:

Primeiro variables globais

```
        //Crear variable de tipo Frm_Principal
        public static Frm_Principal frm_principal = null;

        MyFunc funciones = new MyFunc();

        //Declaramos o buffer de recepción
        byte[] BufferRecepcion = new byte[2];
        //Declaramos o buffer de emisión
        byte[] BufferTransmision = new byte[2];

        int recepcion = 0;
```

Engadimos a liña que habilita a recepción polo porto serie

```
        public Frm_Principal()
        {
            InitializeComponent();
            //Para habilitar a recepción polo porto serie
            Puerto.DataReceived += new
System.IO.Ports.SerialDataReceivedEventHandler(Recepcion);
        }
```

Neste caso, iniciamos o fondo de escala da gráfica a 1000 cando arranca a aplicación.

```
        private void Frm_Principal_Load(object sender, EventArgs e)
        {
            Grafica.get_YAxis(0).Span = 1000;
        }
```

Os eventos de recepción e actualización permiten recoller a información procedente do autómata e visualizala na gráfica

```
        private void Recepcion(object sender, System.IO.Ports.SerialDataReceivedEventArgs e)
        {
            //Lemos bytes
            Puerto.Read(BufferRecepcion, 0, (int)(BufferRecepcion.Length));
            //Chamamos ao evento de actualización
            this.Invoke(new EventHandler(Actualizar));
        }

        private void Actualizar(object s, EventArgs e)
        {
            //Recompoñemos a palabra a partir dos dous bytes
            recepcion = BufferRecepcion[1] * 256 + BufferRecepcion[0];
            Temp0.Position = recepcion;

            //Visualizamos na gráfica
            Grafica.get_Channel(0).AddYElapsedSeconds(recepcion);
        }
```

Temos tamén a opción do menú para chamar ao formulario de configuración.

```
        private void Btn_Config_Click(object sender, EventArgs e)
        {
            frm_principal = this;
            //Creamos en memoria el segundo formulario
            Frm_Secundario frm_secundario = new Frm_Secundario();
            //Amosamos de xeito modal o segundo formulario
            frm_secundario.ShowDialog();
        }
```

A opción para saír do programa.

```
        private void Btn_Salir_Click(object sender, EventArgs e)
        {
            DialogResult aviso = MessageBox.Show("Quere pechar o programa?", "Atenciçon",
                MessageBoxButtons.OKCancel, MessageBoxIcon.Warning);
            if(aviso == DialogResult.OK)
            {
                Puerto.Close();
                this.Close();
            }
        }
```

O evento que enviará o dato ao autómata cando cambie a posición do potenciómetro

```
        private void Poten_OnPositionChangeFinished(object sender, EventArgs e)
        {
            //Neste caso enviamos cada vez que cambie o valor do potenciómetro
            if (Puerto.IsOpen)
            {
                BufferTransmision[1] = Convert.ToByte(Math.Floor(Math.Round(Poten.Position)
/ 256));
                BufferTransmision[0] = Convert.ToByte(Math.Round(Poten.Position) % 256);

                Puerto.Write(BufferTransmision, 0, (int)(BufferTransmision.Length));
            }
        }
```

E, por último, o botón para borrar a gráfica se o desexamos.

```
private void Btn_Borrar_Click(object sender, EventArgs e)
{
    //Borra a gráfica
    Grafica.ClearAllData();
    //Posiciona a gráfica ao principio
    Grafica.get_Channel(0).ResetStartTimeOnFirstDataPoint = true;
}
```